DEVELOPMENT OF ITERATIVE TECHNIQUES FOR THE SOLUTION OF UNSTEADY COMPRESSIBLE VISCOUS FLOWS

Grant NAG-1-1217
Supplement 2

Final Report

Submitted to

NASA Langley Research Center
Hampton, VA 23665

Attn: Dr. Woodrow Whitlow
Chief, Unsteady Aerodynamics Branch

Prepared by

Lakshmi N. Sankar and Duane Hixon
School of Aerospace Engineering
Georgia Institute of Technology, Atlanta, GA 30332

November 1993

SUMMARY

The work done under this project was documented in detail as the Ph. D. dissertation of Dr. Duane Hixon. It may be recalled that the objectives of the research project were:

1) Evaluation of the Generalized Minimum Residual method (GMRES) as a tool for accelerating 2-D and 3-D unsteady flows, and

2) Evaluation of the suitability of the GMRES algorithm for unsteady flows, computed on parallel computer architectures.

Both these objectives were met.

In addition to the Ph. D. dissertation of Mr. Duane Hixon, the following three AIAA papers were pulished, under the present work. Two of these papers also appeared as journal articles.

1. Hixon, R. and Sankar, L. N., " Application of a Generalized Minimum Residual Method to 2-D Unsteady Flows," AIAA Paper 92-0422; also, AIAA Journal, Volume 31, No. 10, October 1993, pp 1955-1957.

2. Hixon, R., Tsung, Fu-Lin and Sankar, L. N., "A Comparison of Two methods for Solving 3-D Unsteady Compressible Viscous Flows," AIAA paper 93-0537, to appear in AIAA Journal, 1994.

3. Hixon, R. and Sankar, L. N., "Unsteady Compressible Two-Dimensional Calculations on a MIMD Parallel Supercomputer," AIAA paper 94-0757.

The first two publications as well as a detailed final report were previously mailed to the sponsor. The AIAA paper 94-0757 is enclosed here, as an appendix.

APPENDIX

AIAA 94-0757

Unsteady Compressible 2-D Flow Calculations on a MIMD Parallel Supercomputer

Duane Hixon and Lakshmi N. Sankar
School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, GA 30332

32nd Aerospace Sciences Meeting & Exhibit
January 10-13, 1994 / Reno, NV

Unsteady Compressible 2-D Flow Calculations on a MIMD Parallel Supercomputer

Duane Hixon* and Lakshmi N. Sankar**
School of Aerospace Engineering
Georgia Institute of Technology
Atlanta, GA 30332

ABSTRACT

An existing sequential 2-D Alternating Direction Implicit (ADI) unsteady compressible viscous flow solver has been modified to run on an Intel iPSC/860 parallel supercomputer. Techniques for implementation of boundary conditions, and the inversion of the implicit matrix equations are discussed. A Generalized Minimal Residual method is added to the parallel algorithm and tested. Results are presented for steady viscous flow past a NACA 0012 airfoil at 13.4 degree angle of attack, and for a NACA 64-A010 airfoil performing a sinusoidal plunging motion under transonic flight conditions. It concluded that implicit time marching algorithms may be efficiently implemented on parallel message passing architectures.

INTRODUCTION

During the past two decades, there has been significant progress in numerical simulation of unsteady compressible viscous flows. At present, a variety of solution techniques exist such as the transonic small disturbance analyses (TSD) [1,2,3], transonic full potential equation based methods[4,5,6], unsteady Euler solvers[7,8], and unsteady Navier-Stokes solvers[9,10,11,12]. This progress has been driven by developments in three areas: (1) Improved numerical algorithms, (2) Automation of body-fitted grid generation schemes, and (3) Advanced computer architectures with vector processing and parallel processing features.

Despite these advances, numerical simulation of unsteady viscous flows still remains a computationally intensive problem even in two dimensions. For example, unsteady 3-D Navier-Stokes simulations of a helicopter rotor blade in forward flight may require over 30,000 time steps for a full revolution of the rotor[10]. In other unsteady flows, such as the high angle of attack flow past fighter configurations, a systematic parametric study of the flow is currently not practical due to the very large CPU time necessary for such simulations[13]. Thus, it is clear that significant improvements to the existing algorithms, and dramatic improvements in computer architectures, will be needed before unsteady two and three dimensional viscous flow analyses become practical day-to-day engineering tools.

One numerical scheme that has been of recent interest is the Generalized Minimal RESidual (GMRES) method originally proposed by Saad and Schultz[14]. This procedure uses a conjugate gradient method to accelerate the convergence of existing flow solvers. GMRES was added to existing steady flow solvers by Wigton, Yu, and Young[15], and has been used on many different types of solvers[16-21]. Saad has also used a similar Krylov subspace projection method on a steady, incompressible Navier-Stokes problem and an unsteady one-dimensional wave propagation equation. The present researchers have successfully used the GMRES scheme to accelerate 2-D and 3-D unsteady viscous flow computations on vector supercomputers[23,24].

In the area of improved computer architectures, emphasis has shifted towards the use of multiple processors. Four different strategies have been pursued by the computer designers. On the Cray Y/MP class of systems, a relatively few sophisticated CPU units tightly connected to each other are used. On massively parallel computers of the CM-5 class, several thousand (relatively) simple processors tightly connected to each other are used. On machines of the Intel iPSC/860 class, a small number of processors (32 or more) tightly connected to each other are used. Finally distributed systems, where a collection of heterogeneous systems connected to each other via standard Ethernet interface lines are coming of age, and rely on a combination of software (e.g. Parallel Virtual Machine interface) and hardware (faster CPUs, high speed communication links) to achieve increased throughput.

---

* Postdoctoral Researcher, Presently at NASA Lewis Research Center, Member AIAA.

** Professor, School of Aerospace Engineering. Senior Member AIAA.

In this work, the GMRES scheme is considered as a candidate for the acceleration of an iterative time marching scheme for unsteady 2-D compressible flow calculations on an Intel iPSC/860 parallel supercomputer. In the past, researchers have ported a number of steady applications to this machine. In these applications, the flow field is divided into a number of blocks, and each CPU node is tasked with advancing the flow field by one pseudo-time step. However, porting a true unsteady flow solver to this machine introduces new difficulties. For example, the practice of lagging boundary conditions at the block boundaries can create serious phase errors in unsteady flow simulations, particularly if large disturbances such as shock waves and strong vortices move across the block boundaries. Most steady flow solvers use an explicit time marching scheme, and the individual blocks (and the CPU nodes) are only loosely coupled to each other. In unsteady viscous flows, implicit schemes are commonly used because of their superior stability characteristics. Unfortunately, implicit schemes require a tight coupling between the blocks, and the CPU nodes. Unless an implicit algorithm is carefully designed, the I/O penalties associated with the data transfer between the nodes can make an implicit algorithm unsuitable for parallel implementation.

MATHEMATICAL AND NUMERICAL FORMULATION

The unsteady, 2-D, compressible ADI code[23] used in this study solves the Navier-Stokes equations, given in curvilinear coordinates as:

$$q_t + E_\xi + G_\zeta = S_\xi + T_\zeta \tag{1}$$

with an implicit scheme similar to that of Steger[25]. Here, **q** is the flow properties vector; **E** and **G** contain the information regarding the mass, momentum and energy fluxes; **S** and **T** contain the viscous stress, heat conduction and viscous work effects. Second or fourth order central differences are used for the spatial derivatives, and a first order backward difference is used for the time derivative.

An iterative ADI scheme is used to numerically integrate the Navier-Stokes equations. At each time step, the following equation is iteratively solved:

$$[I + \Delta\tau\, \delta_\xi A]^{n+1,k}\, [I + \Delta\tau\, \delta_\zeta C]^{n+1,k} \{\Delta q\} = R$$

where, **R** is the residual being driven to zero, given by

$$R = -\Delta\tau\, [(q^{n+1,k} - q^n)/\Delta t] + \delta_\xi (S - E) + \delta_\zeta (T - G)]^{n+1,k} \tag{2}$$

and Δq is the change in the flow properties between successive iterations,

$$\Delta q = q^{n+1,k+1} - q^{n+1,k} \tag{3}$$

Also, 'n' refers to the time level, and 'k' to the iteration level. The time step is given as Δt, while $\Delta\tau$ is a local time step used in the iterative process. The matrices **A** and **C** are the Jacobians of the flux vectors **E** and **G**, and are computed using the information at the previous iteration level 'k'.

Since the left side of Eq. (2) is invertible, this equation may formally be written as:

$$q^{n+1,k+1} = F(q^{n+1,k}) \tag{4}$$

A non-iterative ADI scheme simply means that only one iteration step is performed at each time level.

Note that the right side of Eq. (2) can be computed by a variety of methods: finite volume, finite element, finite difference, etc. The left side ADI matrix may be replaced by an LU form, or even a simple diagonal form. The GMRES formulation does not concern itself with such details of the implementation, but instead treats the flow solver as a 'black box' used only to evaluate Eq. (4).

Of course, the success of the GMRES solver will depend on the left side matrix chosen. Implicit formulations such as ADI and LU schemes are known to perform significantly better than explicit forms[26].

The unsteady GMRES solver attempts to find the $q^{n+1,k}$ that will minimize the equation:

$$M(q^{n+1,k}) = q^{n+1,k+1} - q^{n+1,k} = 0 \tag{5}$$

at each time step.

In a non-iterative ADI scheme, the time step is restricted to prevent errors introduced by the

linearization of the flux terms and the approximate factorization of the linear matrix operator from affecting the unsteady solution. Since the GMRES method is being used on an iterative time marching scheme, the time step is now dependent only on the ability of the discretized equations to follow the physics of the flow. A reduction in CPU time is achieved if the GMRES method requires fewer evaluations of the residual **R** to arrive at a given time level than the non-iterative ADI methodology to arrive at the same time level.

The GMRES solver works by assuming that the error vector $\mathbf{M}(\mathbf{q}^{n+1,k})$ is spanned by a small set of orthonormal direction vectors. For two dimensional compressible flows, there are a total of (imax x kmax x 4) possible orthonormal directions that the correction vector to the flow variables, $\Delta\mathbf{q}$ may lie in.

The GMRES solver uses the underlying iterative ADI scheme to choose a small set of orthogonal directions (a user input, which is usually less than 20). The slope of the residual in each direction is numerically computed. From this, a least squares problem is solved to minimize the magnitude of the l_2 norm of the error vector $\mathbf{M}(\mathbf{q}^{n+1,k})$.

In contrast, the classical iterative ADI scheme given in equation (2) considers only one direction in each iteration; each new iteration computes a new direction which has no relation to the directions that have already been used. This causes, in many cases, the iterative process to stall, and no further reduction in the error vector $\mathbf{M}(\mathbf{q}^{n+1,k})$ is achieved.

Closely following the development given in Ref. 15, the direction vectors are found as follows:

The initial direction is computed as

$$\vec{\mathbf{d}}_1 = \mathbf{M}(\vec{\mathbf{q}}^{n+1,k}) \tag{6}$$

and normalized as

$$\vec{\mathbf{d}}_1 = \frac{\vec{\mathbf{d}}_1}{\|\mathbf{d}_1\|} \tag{7}$$

To compute the remaining directions (j=1,2,...J-1), take

$$\vec{\mathbf{d}}_{j+1} = \overline{\mathbf{M}}(\vec{\mathbf{q}}^{n+1,k};\vec{\mathbf{d}}_j) - \sum_{i=1}^{J} \mathbf{b}_{ij}\vec{\mathbf{d}}_i \tag{8}$$

where

$$\mathbf{b}_{ij} = (\overline{\mathbf{M}}(\vec{\mathbf{q}}^{n+1,k};\vec{\mathbf{d}}_j),\vec{\mathbf{d}}_i) \tag{9}$$

and

$$\overline{\mathbf{M}}(\vec{\mathbf{q}};\vec{\mathbf{d}}_j) = \frac{\mathbf{M}(\vec{\mathbf{q}} + \varepsilon\vec{\mathbf{d}}_j) - \mathbf{M}(\vec{\mathbf{q}})}{\varepsilon} \tag{10}$$

Here, ε is taken to be some small number. In this work, it is set to 0.001.

The new direction $\vec{\mathbf{d}}_{j+1}$ is normalized before the next direction is computed:

$$\mathbf{b}_{j+1,j} = \left\|\vec{\mathbf{d}}_{j+1}\right\| \tag{11}$$

and

$$\vec{\mathbf{d}}_{j+1} = \frac{\vec{\mathbf{d}}_{j+1}}{\mathbf{b}_{j+1,j}} \tag{12}$$

After obtaining the components of $\Delta\mathbf{q}$ along these directions, the solution vector is updated using

$$\vec{\mathbf{q}}^{n+1,k+1} = \vec{\mathbf{q}}^{n+1,k} + \sum_{j=1}^{J} \mathbf{a}_j\vec{\mathbf{d}}_j \tag{13}$$

where the coefficients $\mathbf{a}_j$ are chosen to minimize:

$$\left|\mathbf{M}(\vec{\mathbf{q}}^{n+1,k+1})\right|^2 - \left\|\mathbf{M}(\vec{\mathbf{q}}^{n+1,k} + \sum_{j=1}^{J} \mathbf{a}_j \vec{\mathbf{d}}_j)\right\|^2$$

$$\approx \left\|\mathbf{M}(\vec{\mathbf{q}}^{n+1,k}) + \sum_{j=1}^{J} \mathbf{a}_j \overline{\mathbf{M}}(\vec{\mathbf{q}}^{n+1,k}; \vec{\mathbf{d}}_j)\right\|^2 \qquad (14)$$

PARALLEL IMPLEMENTATION

The GMRES code has been extensively evaluated for a number of unsteady flows in two- and three-dimensions[23,24] .These earlier calculations done were on vector machines of the Cray Y/MP class, or on advanced workstations. The thrust of the present study is to evaluate if the GMRES algorithm performs well on parallel machines of the Intel iPSC/860 class, and determine any modifications needed to tune the algorithm to these machines.

The iPSC/860 is a MIMD machine; the individual processors work with different subsets of the data simultaneously, and each processor may independently execute different instructions at the same time. Furthermore, the iPSC/860 is a distributed-memory machine, where each processor has its own separate memory. In order to obtain information from another processor, a message-passing routine must be explicitly coded. Since the message passing process is a relatively slow sequential process, the most efficient code will usually have the least number of messages.

As a first step, a non-iterative 2-D ADI code (that solves equation (2), but uses only one iteration) was modified to run on an Intel iPSC/860 machine located at the NASA Langley Research Center. In order to accomplish this goal, the computational domain was divided into a number of blocks or sub-domains as shown in Figure 1. As is common with block structured grid solvers, each sub-domain overlaps its neighbors by two "ghost" cells. Each processor performs an ADI step over one or more blocks. The boundary conditions for the ghost cells are updated by passing messages at the end of each step.

A number of 2-D steady flow calculations were first carried out with the non-iterative ADI solver implemented on the iPSC/860 architecture. The steady state solutions were identical to the results obtained on sequential machines. As stated earlier, this approach of lagging the flow properties at the block boundaries (ghost cells) works well only for steady flows.

Iterative Thomas Algorithm

Since the goal of this work was to solve the unsteady Navier-Stokes equations in a time-accurate fashion, an iterative solution procedure was implemented. The bottleneck in such an implementation is inversion the tridiagonal matrix system in the streamwise (ξ -) direction.

It should be noted that a parallel ADI step is quite different from that of the sequential version. A description of the current implementation follows:

First, the order of the sweeps is reversed:

$$[\mathbf{I} + \Delta\tau\partial_\zeta \mathbf{C}][\mathbf{I} + \Delta\tau\partial_\xi \mathbf{A}]\{\Delta\mathbf{q}\} = \{R^{n+1,k}\} \qquad (15)$$

and the ζ-sweep is performed first.

$$[\mathbf{I} + \Delta\tau\partial_\zeta \mathbf{C}]^{n+1,k}\{\Delta\mathbf{q}^*\} = \{R^{n+1,k}\} \qquad (16)$$

Since this sweep requires information only within a given block, and never requires information across block boundaries, this matrix inversion was done, in parallel on all the processors, using the Thomas algorithm in a manner identical to the sequential version of the flow solver.

When equation (16) has been solved in all the blocks by all the CPU nodes, the values for $\Delta\mathbf{q}^*$ are known throughout the flow field. Next, the streamwise sweep is performed:

$$[\mathbf{I} + \Delta\tau\partial_\xi \mathbf{A}]^{n+1,k}\{\Delta\mathbf{q}\} = \{\Delta\mathbf{q}^*\} \qquad (17)$$

This sweep, performed using Thomas algorithm, usually requires information across block boundaries, because the nodes along the ξ- direction in all the blocks are implicitly coupled. In the iterative approach we lagged the $\Delta\mathbf{q}$ values at the block boundaries by one iteration, or set these values to zero. This strategy removes the implicit coupling between the individual blocks.

We found the above approach to be unsatisfactory for a number of reasons. A large number of iterations were needed to drive the Δq values, and the associated phase errors to zero. This algorithm required two message passes per iteration for each processor, which increases the run time dramatically. Also, the convergence of the Δq values near the block boundaries was not uniform between iterations.

Block Cyclic Reduction (BCR) Routine

To avoid the difficulties involved in the iterative Thomas algorithm, a Block Cyclic Reduction (BCR) routine was next implemented, for solving equation (17) in the ξ-sweep.

While the Thomas algorithm requires the least operation count to solve the matrix system , it also is an inherently sequential method (i.e., for each step in the inversion procedure, information is required from the step previously performed). Therefore, the Thomas algorithm is not directly parallelizable.

The Block Cyclic Reduction routine is a more efficient way of solving the tridiagonal matrix equations. Given a tridiagonal matrix that is $(2^n+1) \times (2^n+1)$, this procedure directly solves the matrix system as described in Ref. 27 .

The BCR routine has three drawbacks. First, the processors must communicate before every round of reduction and back-substitution to obtain matrix values that lie outside its block. These messages are relatively short, however. Second, during the end of the reduction and the beginning of the back-substitution process when there are few lines left to compute, several processors wait in idle. It was anticipated that the savings in computation time compared to the parallel iterative routine will make up for the idle time encountered. Third, for best performance, the BCR routine must have $2^n + 1$ equations to solve. Thus, the grid required for this routine is less flexible than that for the iterative parallel code.

It should be emphasized that the BCR routine is a direct solution procedure in this implementation. There is no need to lag the Δq values at the block boundaries, as required by the iterative Thomas algorithm described earlier.

GMRES Implementation

The GMRES routines were finally added to the parallel iterative ADI code after the ADI code was validated. When the GMRES algorithm was implemented, a question arose as to the definition of the residual to be minimized.

Two ideas were tried. The first idea was a completely parallel GMRES implementation, where each processor ran a GMRES routine to minimize the l_2 norm of the error vector **M** for nodes only in its particular block. When a function evaluation is required, the processors work in parallel to compute the residual **R** and the error vector **M** in all the blocks. The GMRES routine on each processor is only concerned with minimizing the error vector in its own block. This is equivalent to allowing each direction to have a different weighting coefficient in each block.

The second idea was a global GMRES implementation. In this scheme, the processors work as before to compute the search directions and the residual in its sub-domain, but at the end of each function evaluation, the global norm of the error vector **M** is computed and used. This is now directly equivalent to the sequential GMRES code in that a single weighting coefficient is used for each direction throughout the flow field.

Initial tests showed that the global GMRES implementation performed significantly better than the local GMRES; thus, the global GMRES was used for all runs. The GMRES algorithm was implemented on both the Thomas and BCR versions of the code. The number of search directions were limited to 5 due to memory limitations.

RESULTS AND DISCUSSIONS

The parallel ADI code was implemented on the NASA Langley 32 processor Intel iPSC/860 MIMD parallel supercomputer. Both the iterative Thomas algorithm and the BCR versions were extensively tested. The BCR solver was typically four times faster than the iterative Thomas algorithm. Here we document only the calculations with the Block Cyclic Reduction method.

The notation used for the GMRES discussion is as follows. The notation ' GMRES(5) ' refers to a steady flow calculation with 5 search directions used at each step. The notation ' GMRES (5/10) ' refers to an unsteady flow calculation with 5 search directions used at each time step; each time step, however, is 10 times that taken by the non-iterative ADI solver.

The steady flow validation case was that of a subsonic viscous flow about a NACA 0012 airfoil at a 13.4° angle of attack. The freestream Mach number was 0.301, and the Reynolds number was 3,950,000. This case was tested experimentally by McAlister, et. al[28]. A C-grid topology was used, with 259 streamwise points

and 41 normal points. The Baldwin-Lomax turbulence model was used for all viscous calculations.

The non-iterative ADI code was run for 1000 iterations on 4,8,16, and 32 processors, and the speedup obtained is shown in Figure 2 and in Table I below. It can be seen that the speedup is not ideal, but this is largely due to the low number of points on each processor. In other words, the processors spent a significant portion of the time passing boundary condition information from block to block. The I/O time associated with the message passing was large, and comparable to the CPU time for the parallel task of computing the residual R or the error vector M.

From this point on, all results shown are obtained using 8 processors. All GMRES solutions are obtained using 5 search directions per step.

Results for the steady runs are shown in Figures 3, 4, and 5. Figure 3 shows the global residual histories for the ADI solver and the GMRES solver. It should be noted that it was only possible to use 5 search directions due to the memory limitations of the machine; previous tests on a sequential computer have shown that GMRES provides significant speedups with more search directions.

Figure 4 shows the lift coefficient history for this run. Note that the GMRES solver converges to the final lift coefficient much faster than the non-iterative BCR solver.

Figure 5 shows the pressure coefficient computed by both the ADI and GMRES(5) solver compared to the experimental results of McAlister, et. al. Good agreement is obtained with experiment, and the solvers return identical answers, even though the GMRES solver has stalled at a relatively high residual. More search directions would have eliminated this problem.

Next, an unsteady validation case was run. The problem studied was that of inviscid transonic flow about a plunging NACA 64-A010 airfoil. The freestream Mach number is 0.8, and the reduced frequency is 0.2. The plunging motion is defined by this equation:

$$y_\tau = -M_\infty \sin(1^\circ)\sin(\omega\tau) \qquad (18)$$

This is a challenging case for unsteady flow solvers, because of the formation and motion of strong shock waves on the upper and lower surfaces. The shock speed is sensitive to the errors in the discretization, and errors introduced at the block boundaries will be expected to adversely affect the solution accuracy. This case has been studied by many researchers; our results are compared to those of Steger[25] and Magnus[26].

The grid used here is a parabolic C-grid, with 259 streamwise and 21 normal grid points. The non-iterative ADI solver uses a non-dimensional time step of .01.

The GMRES solutions shown are computed using 5 search directions and 5 and 10 times the ADI time step (GMRES (5/5) and GMRES (5/10), respectively). Local time stepping is used as a preconditioner for the GMRES solver; this improves the convergence dramatically.

The results shown are for the fourth cycle of oscillation. Figure 6 shows the lift coefficient histories as a function of the phase angle. It can be seen that the ADI and both GMRES solutions agree well with the earlier studies of Steger and Magnus. Good agreement between different methods for the lift coefficient may be deceptive, because the errors (in shock locations and shock speeds) on the upper and lower surfaces tend to offset each other. The pitching moment is much more sensitive to the shock location, and serves as a more suitable indicator of the solution accuracy.

Figure 7 shows the moment coefficient histories for the same case. It is seen that the ADI and GMRES (5/5) solutions agree well with the earlier studies of Steger and Magnus, while the moment coefficient for the GMRES (5/10) run is not close at all. This is due to the errors in the computed shock speed as a larger time step is used.

In this study we used a simple preconditioner to stabilize the calculations. The time pseudo time step $\Delta\tau$ shown in equation (2) was different from the physical time step Δt. Of course, the pseudo-step does not have any effect on the final converged solution at a given time step. In some earlier studies on a sequential computer, the preconditioner was not used (i.e. $\Delta\tau = \Delta t$). In these earlier calculations, 10 search directions were necessary to stabilize the unsteady GMRES procedure (i.e., GMRES (10/5)), while now 5 are sufficient.

It should be noted that this case is a worst-case scenario for the GMRES solver. On most problems, the GMRES solver (both the parallel implementation and the sequential implementation) can speed up the solution procedure by a factor of 3, over the baseline iterative ADI solver. The present problem, with its moving shocks and nonlinear flow field, is a harsh test of the GMRES solver.

CONCLUSIONS

An existing sequential unsteady 2-D compressible viscous flow solver was rewritten for implementation on an Inter iPSC/860 MIMD parallel supercomputer. Two methods were investigated for the parallel solution of the tridiagonal block matrices encountered in the ADI procedure. the Block Cyclic Reduction technique proved to be four times faster than an iterative Thomas algorithm.

The code was validated for steady and unsteady,, viscous and inviscid flow cases. A GMRES solution method was added and validated, and the effect of preconditioning on the GMRES parallel implementation was investigated. This proved to have a very beneficial effect, and halved the number of search directions originally required for this problem.

The GMRES method proved to be highly parallelizable, requiring only one very short message to be passed for each search direction. The main shortcomings encountered were in the parallelization of the underlying ADI algorithm. Algorithms such as the BCR algorithm described here, or other procedures that exploit the parallel architecture more efficiently than BCR can result in dramatic speedups with the GMRES solver.

ACKNOWLEDGMENTS

The research reported here was supported by a grant from the NASA Langley Research Center (Grant No. NAG-1-1217). Dr. Woodrow Whitlow was the technical monitor.

REFERENCES

1 Borland, C.J. and Rizzetta, D., "Nonlinear Transonic Flutter Analysis," AIAA Paper 81-0608-CP, AIAA Dynamic Specialists Conference, 1981.

2. Rizzetta, D.P. and Borland, C., "Numerical Solution of Unsteady Transonic Flow over Wings with Viscous-Inviscid Interaction," AIAA Paper 82-0352, Jan. 1982.

3. Batina, J.T., "Unsteady Transonic Algorithm Improvements for Realistic Aircraft Applications", Journal of Aircraft, Vol. 26, No. 2, Feb. 1989, p. 131-9.

4. Sankar, L.N., Malone, J.B., and Tassa, Y., "An Implicit Conservative Algorithm for Steady and Unsteady Three Dimensional Transonic Potential Flows", AIAA Paper 81-1016-CP, June 1981.

5. Malone, J.B. and Sankar, L.N., "Application of a Three-Dimensional Steady and Unsteady Full Potential Method for Transonic Flow Computations", AFWAL-TR-84-3011, Flight Dynamics Laboratory, Wright Patterson Air Force Base, Dayton, OH, May 1984.

6. Shankar, V., Ide, H., Gorski, J., and Osher, S., "A Fast, Time-Accurate Unsteady Full Potential Scheme," AIAA Paper 85-1512-CP, July 1985.

7. Pulliam, T.H. and Steger, J.L., "Implicit Finite-Difference Simulations of 3-D Compressible Flow", AIAA Journal, Vol. 18, Feb. 1980, pp. 159-167.

8. Batina, J.T., "Unsteady Euler Solutions Using Unstructured Dynamic Meshes," AIAA Paper 89-0115, Jan. 1989.

9. Sankar, L.N. and Tang, W., "Numerical Solution of Unsteady Viscous Flow Past Rotor Sections," AIAA Paper 85-0129.

10. Wake, B.E. and Sankar, L.N., "Solutions of the Navier-Stokes Equations for the Flow About a Rotor Blade", Journal of the American Helicopter Society, Vol. 34, No. 2, April 1989, pp. 13-23.

11. Rai, M.M., "Navier-Stokes Simulations of Rotor-Stator Interaction Using Patched and Overlaid Grids," AIAA Paper 85-1519-CP, July 1985.

12. Gatlin, B. and Whitfield, D.L., "An Implicit Upwind Finite Volume Scheme for Solving the Three-Dimensional Thin-Layer Navier-Stokes Equations," AIAA Paper 87-1149-CP, June 1987.

13. Kwon, O.J. and Sankar, L.N., "Viscous Flow Simulation of a Fighter Aircraft", Journal of Aircraft, Vol. 29, No. 5, Sep-Oct. 1992, pp. 886-891.

14. Saad, Y. and Schultz, M.H, "GMRES: A Generalized Minimum Residual Algorithm for Solving Nonsymmetric Linear Systems", SIAM J. Sci. Stat. Comp., Vol. 7, No. 3, 1986, pp.856-869.

15. Wigton, L. B., Yu, N. J., and Young, D. P., "GMRES Acceleration of Computational Fluid Dynamics Codes", AIAA Paper 85-1494-CP, 1985.

16. Venkatakrishnan, V. and Mavriplis, D. J., "Implicit Solvers for Unstructured Meshes", ICASE Report 91-40, May 1991.

17. Giannakoglou, K., Chaviaropoulos, P., and Papailiou, K., "Acceleration of Standard Full-Potential and Elliptic Euler Solvers, Using Preconditioned

Generalized Minimal Residual Techniques", ASME FED v. 69, Published by ASME, New York, NY, USA. pp. 45-52.

18. Young, D.P., Melvin, R.G., Bieterman, M.B., Johnson, F.T., and Samant, S.S., "Global Convergence of Inexact Newton Methods for Transonic Flow", International Journal for Numerical Methods in Fluids, Vol. 11, No. 8, Dec. 1990, pp. 1075-1095.

19. Vuik, C., "Solution of the Discretized Incompressible Navier-Stokes Equations with the GMRES Method", Netherlands Mathematical Institute REPT-91-24, 1991.

20. Adams, L.M., and Ong, E.G., "Comparison of Preconditioners for GMRES on Parallel Computers", AMD Symposia Series, Vol. 86. Published by ASME, New York, NY, USA., pp. 171-186.

21. Qin, X., and Richards, B.E., "a-GMRES: A New Parallelizable Iterative Solver for Large Sparse Non-Symmetric Linear Systems Arising from CFD", International Journal for Numerical Methods in Fluids, Vol. 15, No. 5, Sep. 15, 1992, pp.113-23.

22. Saad, Y. and Semeraro, B. D. , Application of Krylov Exponential Propagation to Fluid Dynamics Equations", AIAA Paper 91-1567-CP, 1991.

23. Hixon, R. and Sankar, L.N., "Application of a Generalized Minimal Residual Method to 2-D Unsteady Flows," AIAA Paper 92-0422, Jan. 1992.

24. Hixon, R., Tsung, F.L., and Sankar, L.N., "A Comparison of Two Methods for Solving 3-D Unsteady Compressible Flows", AIAA Paper 93-0537, Jan. 1993.

25. Steger, J. L., "Implicit Finite Difference Simulation of Flow about Arbitrary Two-Dimensional Geometries", AIAA Journal, Vol. 16, July 1978, p. 679-86.

26. Magnus, R. and Yoshihara, H., "Unsteady Transonic Flow over an Airfoil', AIAA Journal, Vol. 14, Dec. 1975, pp. 1622-1628.

27. Wake, B.E. and Egolf, T.A., "Implementation of a Rotary-Wing Navier-Stokes Solver on a Massively Parallel Computer", AIAA Journal, Vol. 29, No. 1, Jan. 1991, p. 58-67.

28. McAlister, K.W., Pucci, S.L., McCroskey, W.J., and Carr, L.W., "An Experimental Study of Dynamic Stall on Advanced Airfoil Sections, Volume 2: Pressure and Force Data", NASA TM 84245, Sept. 1982.

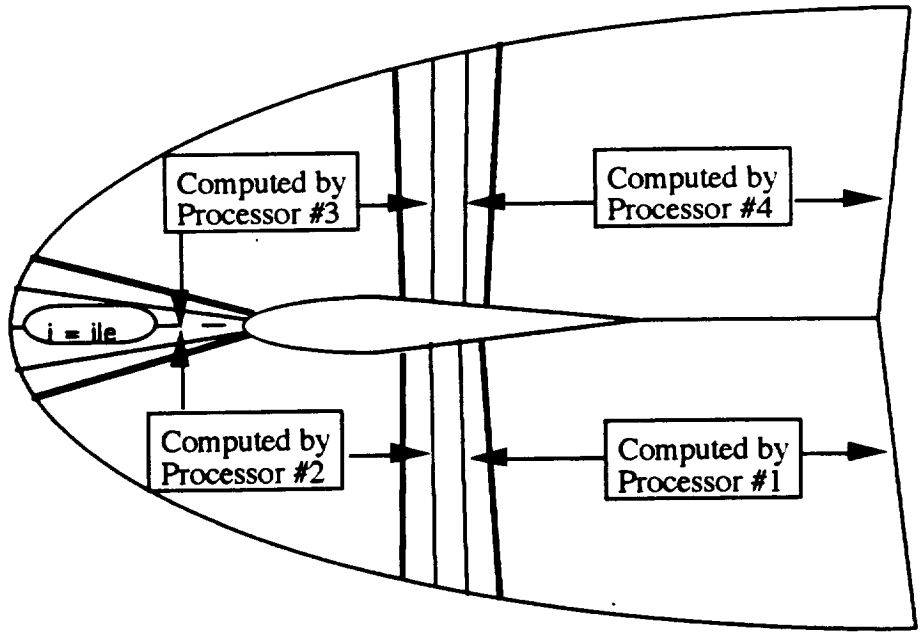

Figure 1. Domain Decomposition for Implementation on a Four Processor Parallel Computer

Number of Processors	CPU time required for 1000 iterations	CPU time for ideal speedup	CPU time required/ideal CPU time
4	6037	6037	1.0
8	3230	3018.5	1.070
16	1985	1509.25	1.315
32	1463	754.625	1.939

Table 1: Parallel Run Timings for 1000 Iterations

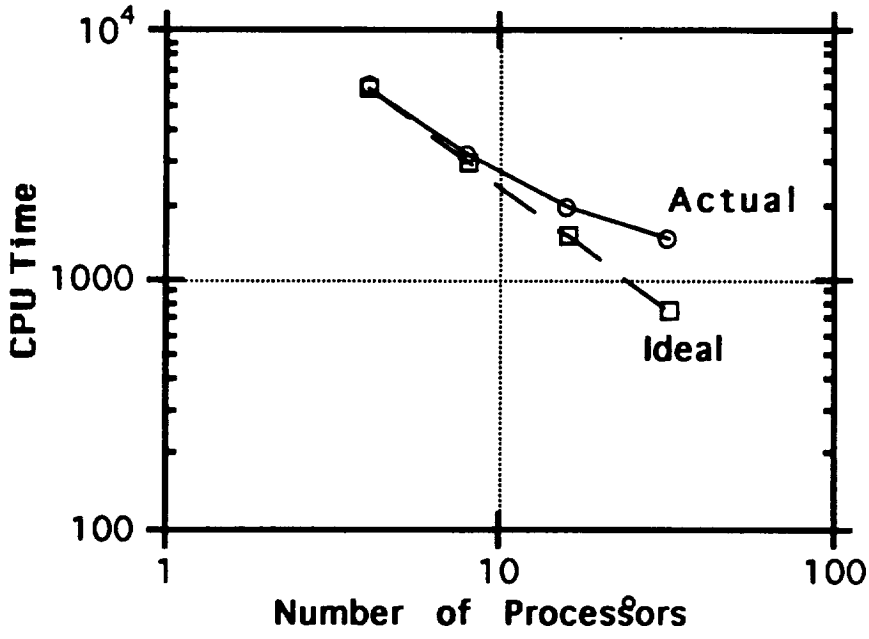

Figure 2: Actual Speedup Compared to Ideal for Viscous BCR Solver using 259 x 41 Grid

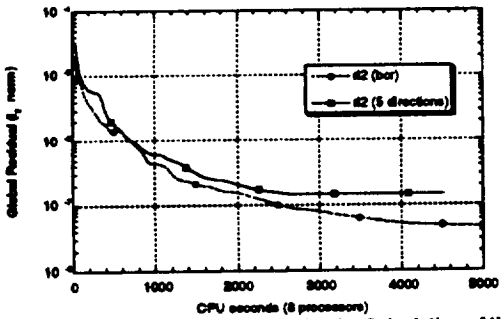

Figure 3: Global Residual History for the Calculation of the Steady Viscous Flow about a NACA 0012 Airfoil (α = 13.4°, M = 0.301, Re = 3,950,000)

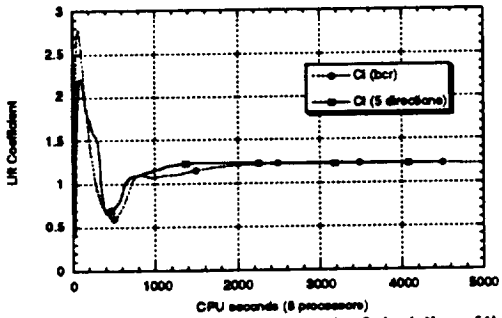

Figure 4: Lift Coefficient History for the Calculation of the Steady Viscous Flow about a NACA 0012 Airfoil (α = 13.4°, M = 0.301, Re = 3,950,000)

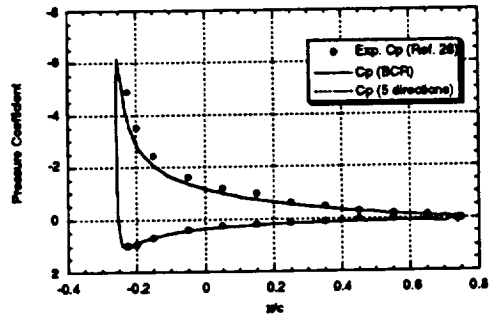

Figure 5: Pressure Coefficient Result for the Calculation of the Steady Viscous Flow about a NACA 0012 Airfoil (α = 13.4°, M = 0.301, Re = 3,950,000)

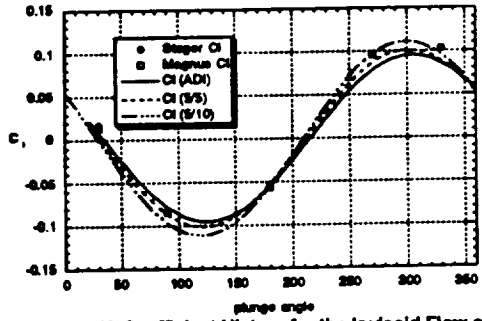

Figure 6: Lift Coefficient History for the Inviscid Flow about a Plunging NACA 64-A010 Airfoil (M = 0.8; k = 0.2)

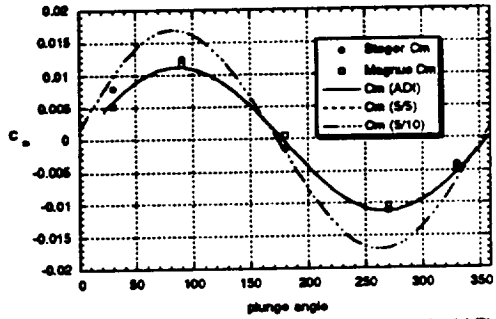

Figure 7: Moment Coefficient History for the Inviscid Flow about a Plunging NACA 64-A010 Airfoil (M = 0.8; k = 0.2)